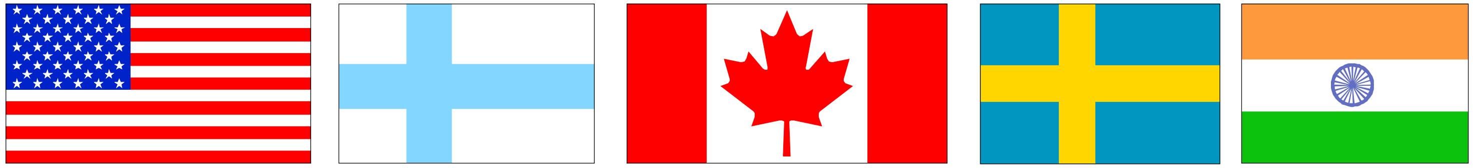
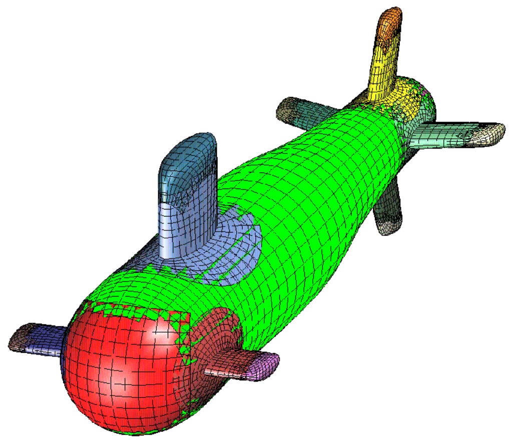
Overture Tools for Geometry Management and Mesh Generation

Kyle Chand
Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, California
www.llnl.gov/CASC/Overture



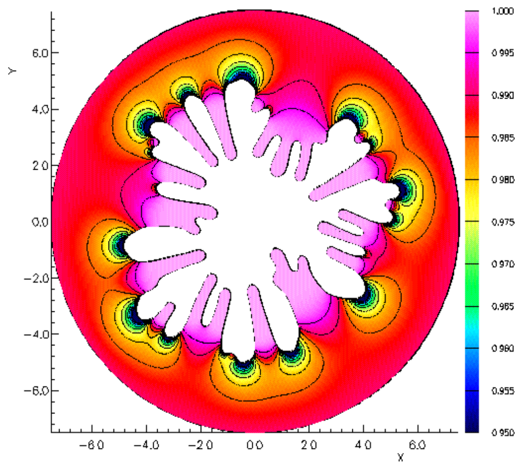
Overture team: David Brown, Kyle Chand, Petri Fast,
Bill Henshaw, Brian Miller, Anders Petersson,
Bobby Phillip, Dan Quinlan

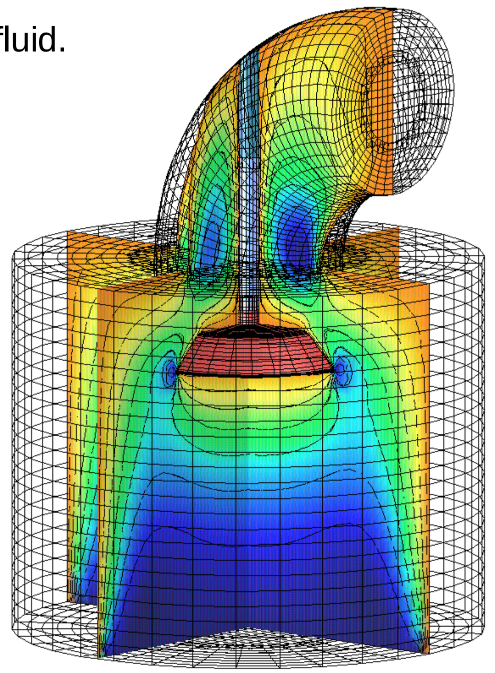
Overture: A Toolkit for Solving PDEs



Overlapping Grids
Fig: Bill Henshaw

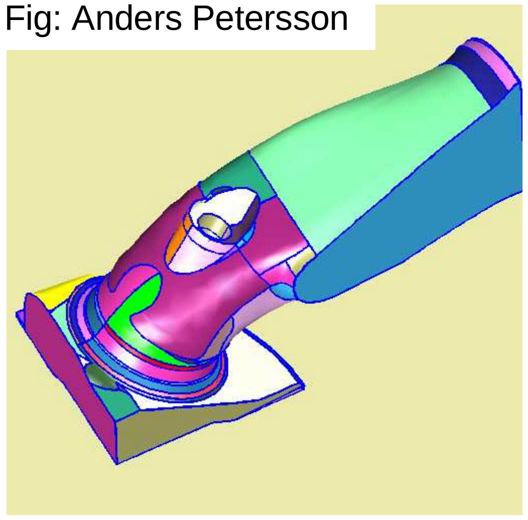
Hele Shaw flow of a non-Newtonian fluid.
Fig: Petri Fast.

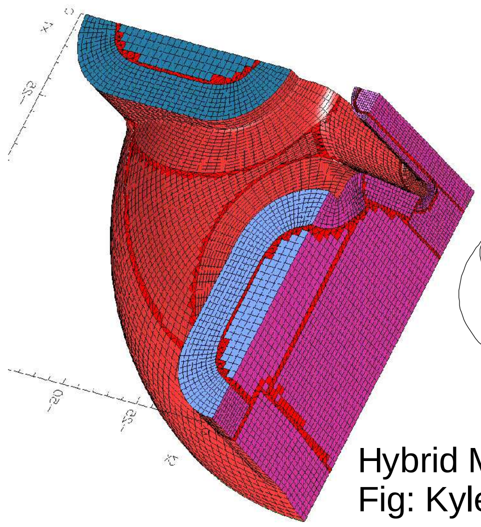




Moving Piston, Incompressible Navier-Stokes
Fig: Bill Henshaw.

CAD Geometry
Fig: Anders Petersson



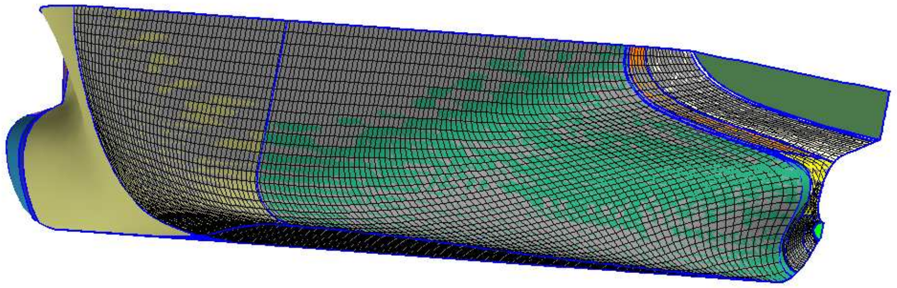


Hybrid Meshes
Fig: Kyle Chand

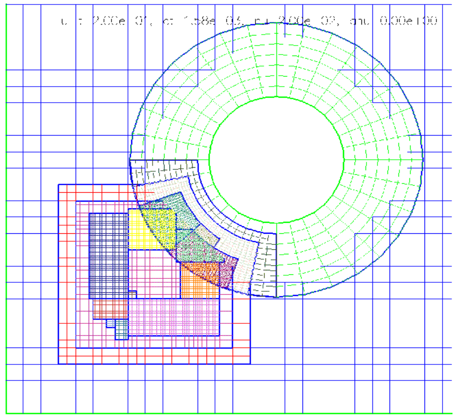
$$\begin{tikzcd}
\boxed{\text{PDE Solver Development}} & & \\
& \boxed{\text{Grid Generation}} & \\
& & \boxed{\text{Geometry}}
\arrow[from=2-2, to=1-1, bend left]
\arrow[from=3-3, to=2-2, bend left]
\arrow[from=3-3, to=1-1, bend left]
\end{tikzcd}$$

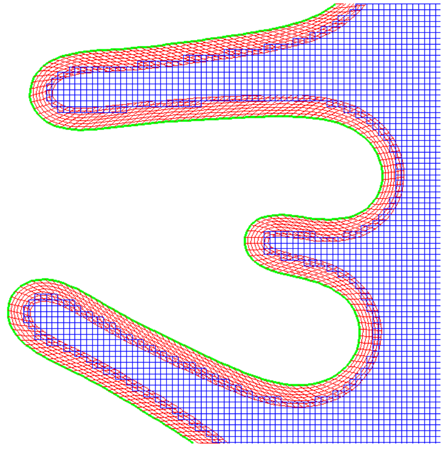
Geometry : mesh generation and PDE discretization

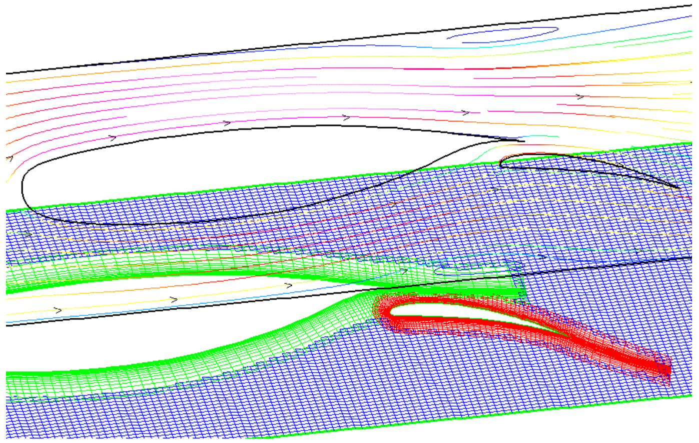
Initial geometry and
grid generation
(surface and volume grids)



Adaptive meshes and
moving/deforming grids
(projection, grid generation, etc)







PDE Discretization:
Mapping derivatives (curvilinear grids)
Higher order Mapping information
Boundary conditions...

Geometry modeling requirements

Creation and importing of geometry (especially from CAD, but not limited to it)

Manipulation operations (intersection, trimming, sectioning, etc)

Fast queries for projection and surface derivatives

Low memory and high performance for geometric queries in simulations (also a nice interface for such codes...)

Low cost (Free!) for us and other researchers

Solution : Implement our own geometry code

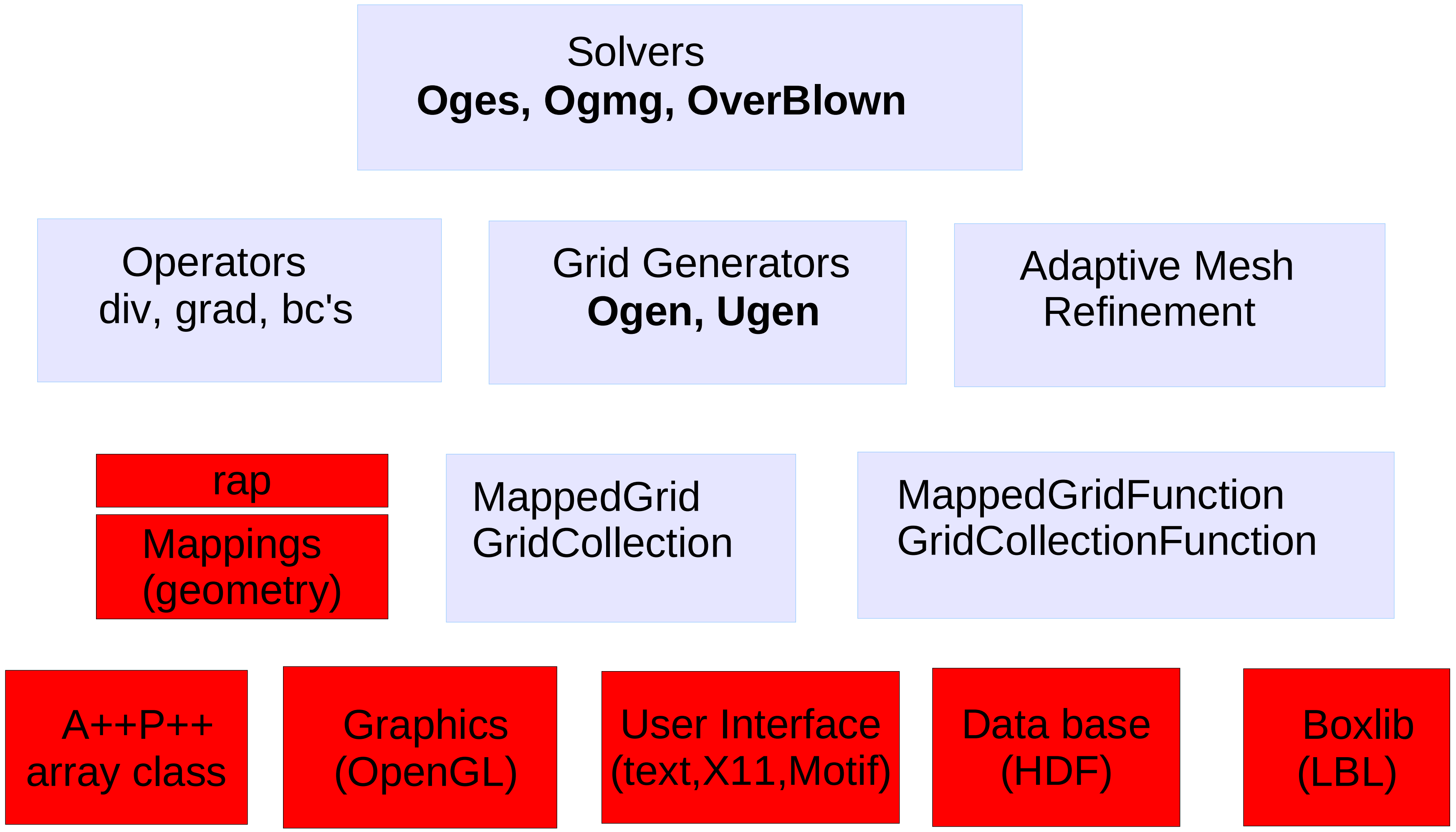
One Option:

Directly interface commercial CAD/Geometry software
+ Robust and accurate interpretation of geometry
+ No need to support the software
- Proprietary and expensive
-/+ Generic interfaces for CAD/3D Geometry creation
- Accuracy is product dependent (translations are poor)
- Efficiency?

Our Solution:

Write the tools we need, give them the interfaces we want
+ Generic to many 3rd party CAD/Geometry tools via neutral files
+ We control the performance and accuracy to meet our needs
+ Open source allows researchers to advance the state of the art
- Translation errors and lost information must be resolved.
-/+ We have to develop and support the code

Overture: A toolkit for solving PDEs

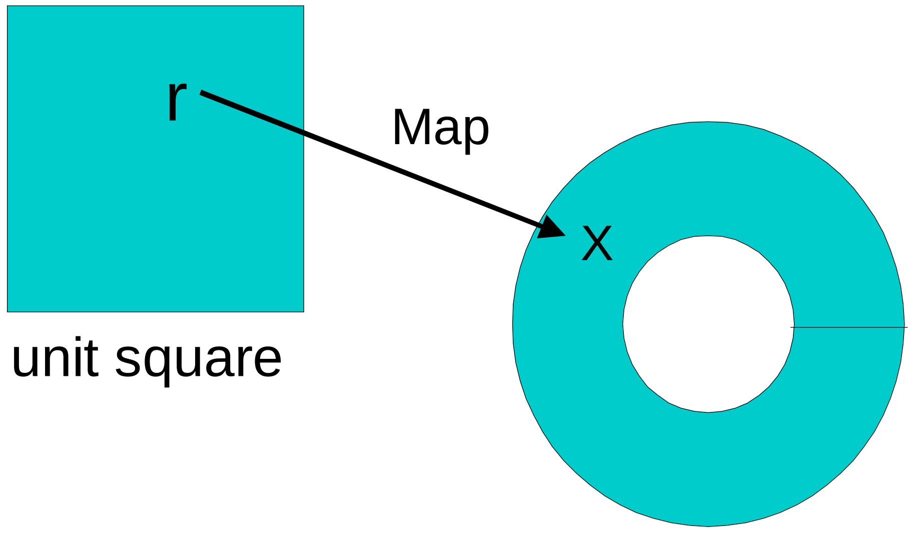


Rapsodi : A geometry toolkit for mesh generation and discretization

Mappings encapsulate the interface to geometry

A Mapping defines a continuous transformation

Each mapping has an optimized ***map*** and *inverseMap*

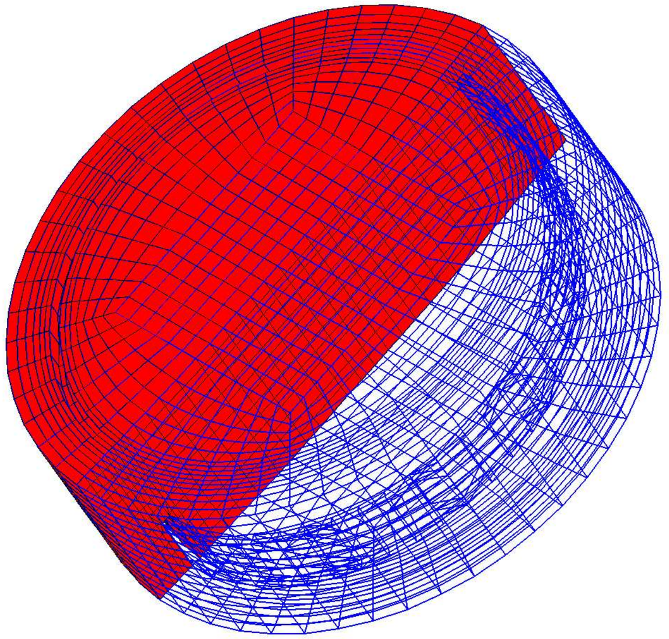


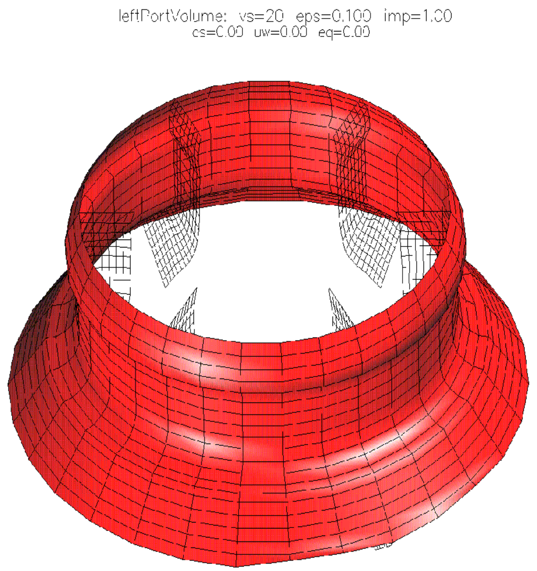
```
class Mapping
{
public:
   virtual void map(...);
   virtual void inverseMap(...);
   virtual int  project(...);

   int   getDomainDimension();
   int   getRangeDimension();
   Bound getRangeBound(...);
   ...
};
```

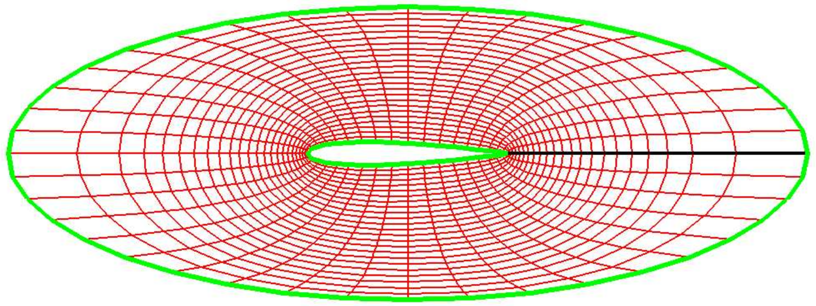
SquareMapping, AnnulusMapping, SphereMapping, HyperbolicMapping, EllipticTransform, MatrixTransform, TrimmedMapping,...

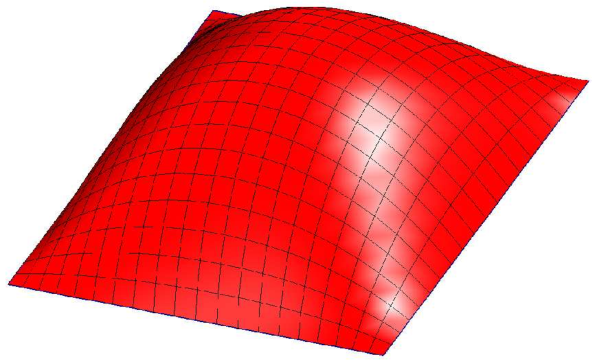
> 40 Mappings

Menagerie of Mappings

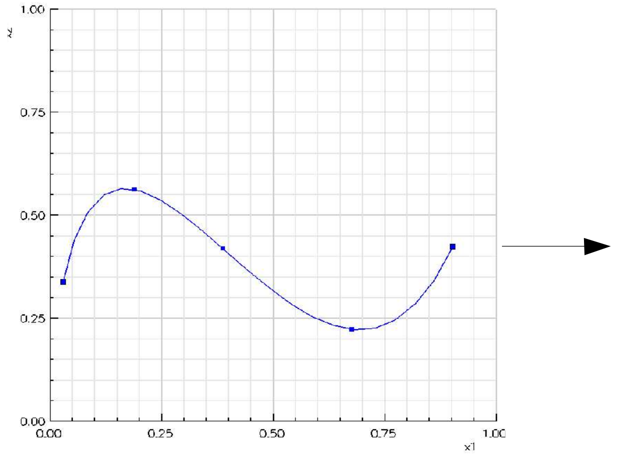




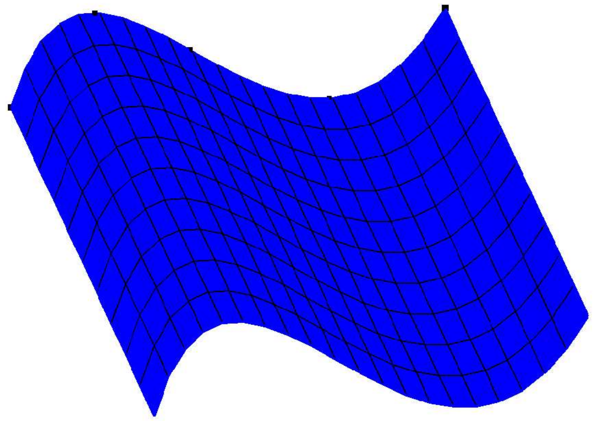




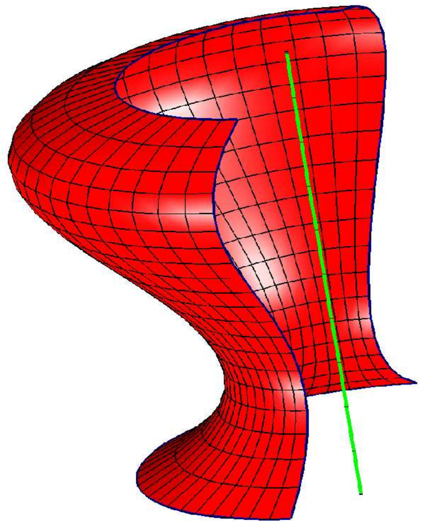
Menagerie of Mappings

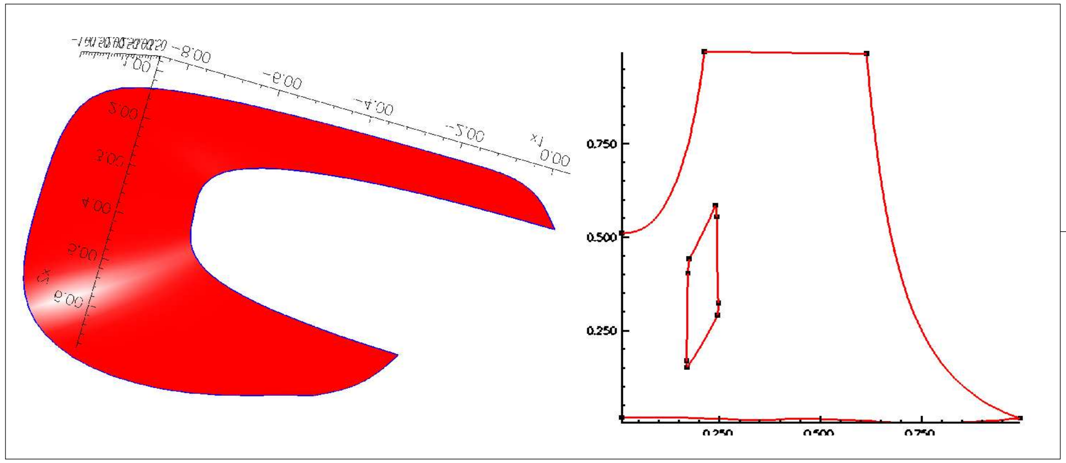


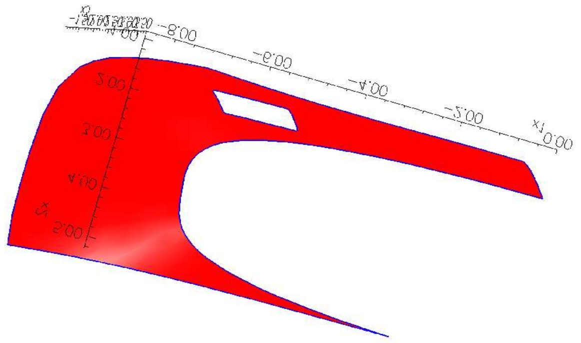
{Depth,Sweep}Mapping



RevolutionMapping







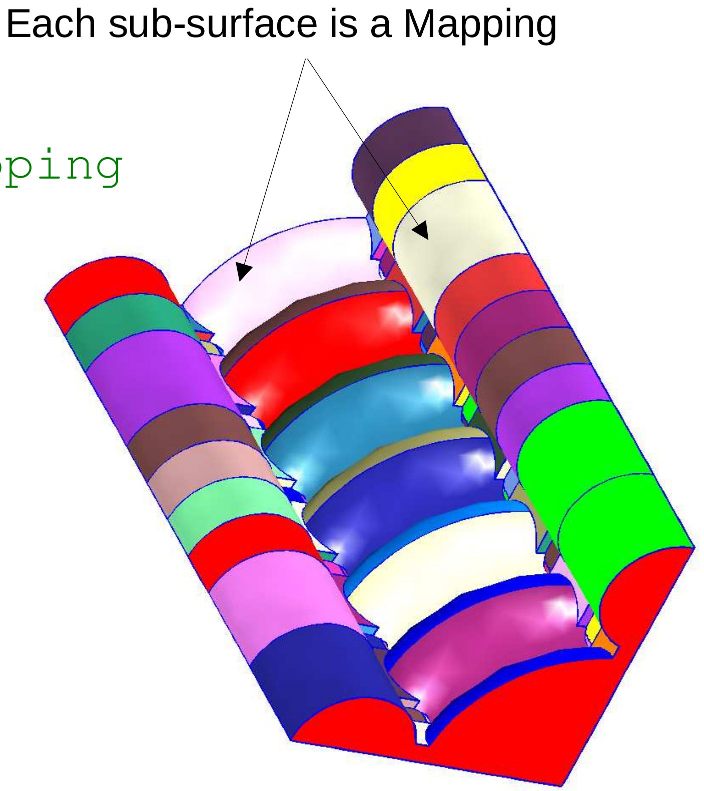
TrimmedMapping

CompositeSurface

Describes geometry using a collection of Mappings

Often obtained from CAD translations:
 can contain too much detail
 are error prone

```
class CompositeSurface : public Mapping
{
public:
   int project(...);

   int numberOfSubSurfaces();
   Mapping & operator[](int);
   int add(...);
   int remove(...);

   CompositeTopology *
      getCompositeTopology();
   ...
};
```



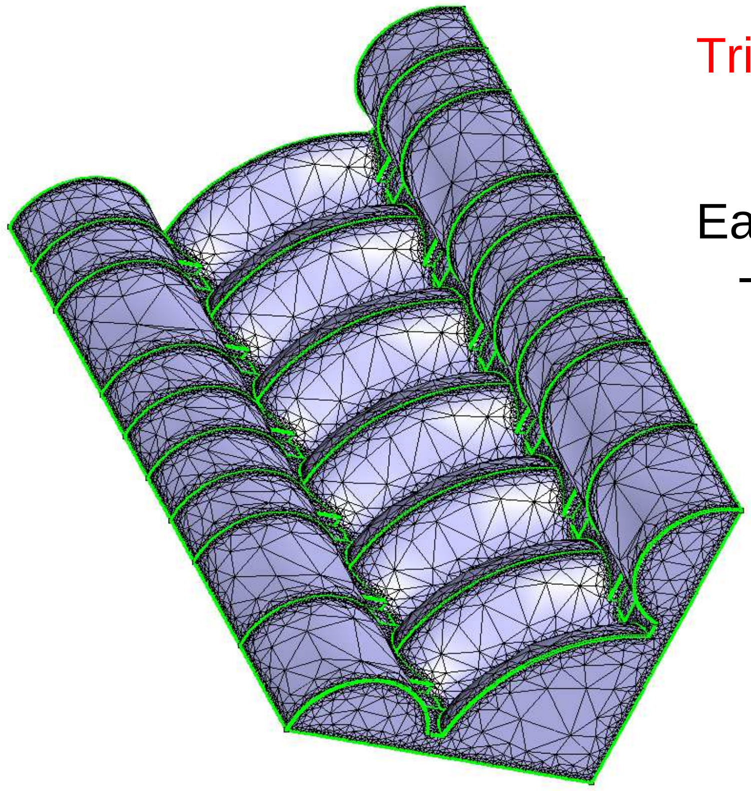
CompositeTopology

Encapsulates the relationship between component surfaces

Automatically detects and corrects small gaps and overlaps

Triangulation provides a fast data structure for projection

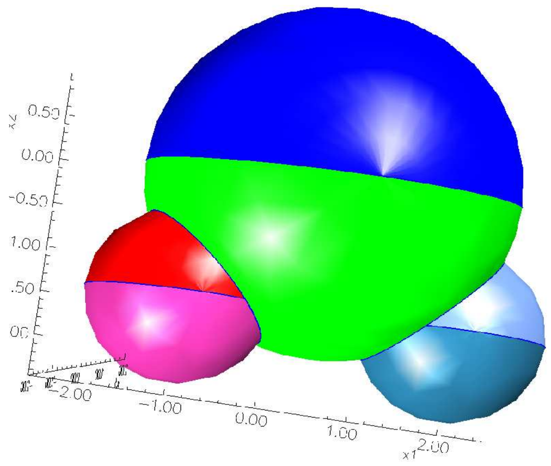
Automatic refinement based on surface deviation
--> useful output to other mesh generators (e.g. cart3d)

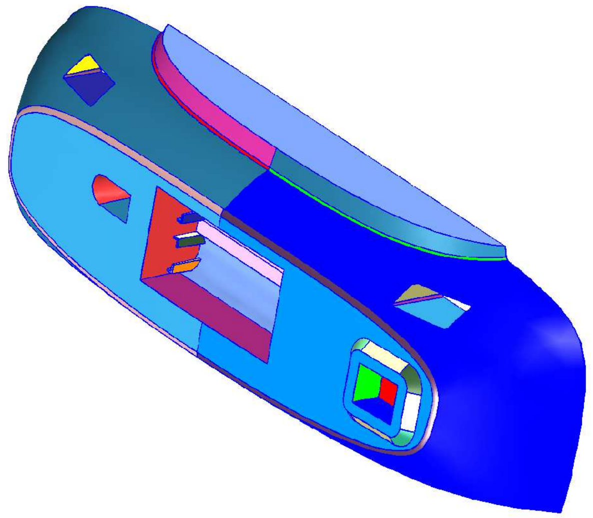


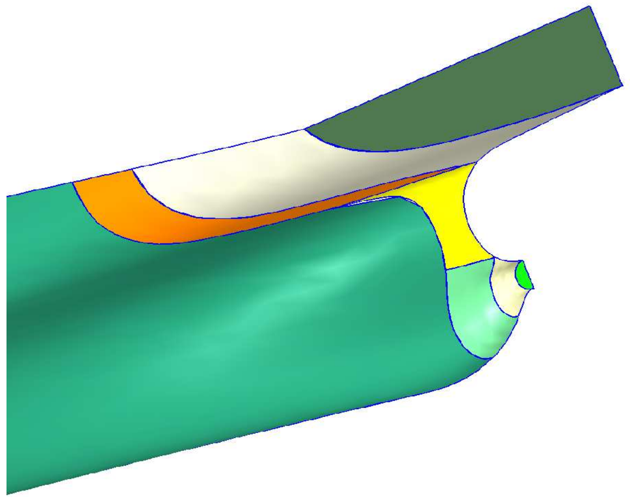
Triangulation is a search data structure,
NOT a computational mesh

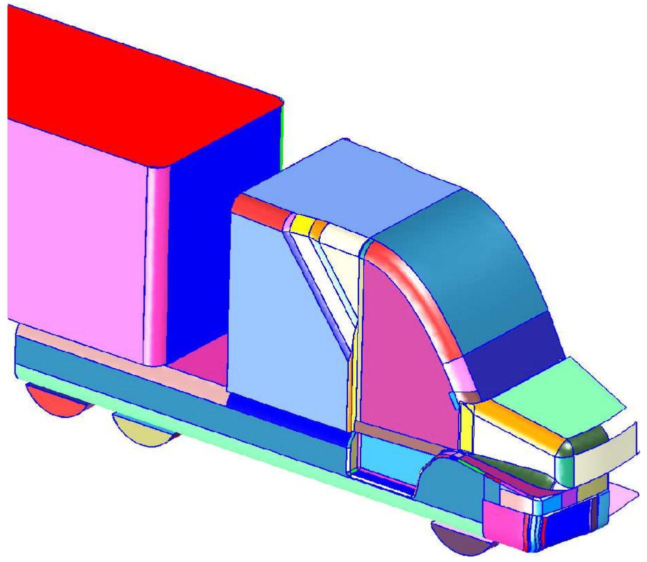
Each triangle maps onto only one sub-surface
--> Fast searches for projections:
-> First project onto triangulation using
a geometric search tree or a walk
from a previously cached triangle
-> Then project onto the
triangle's sub surface

Composite Surfaces

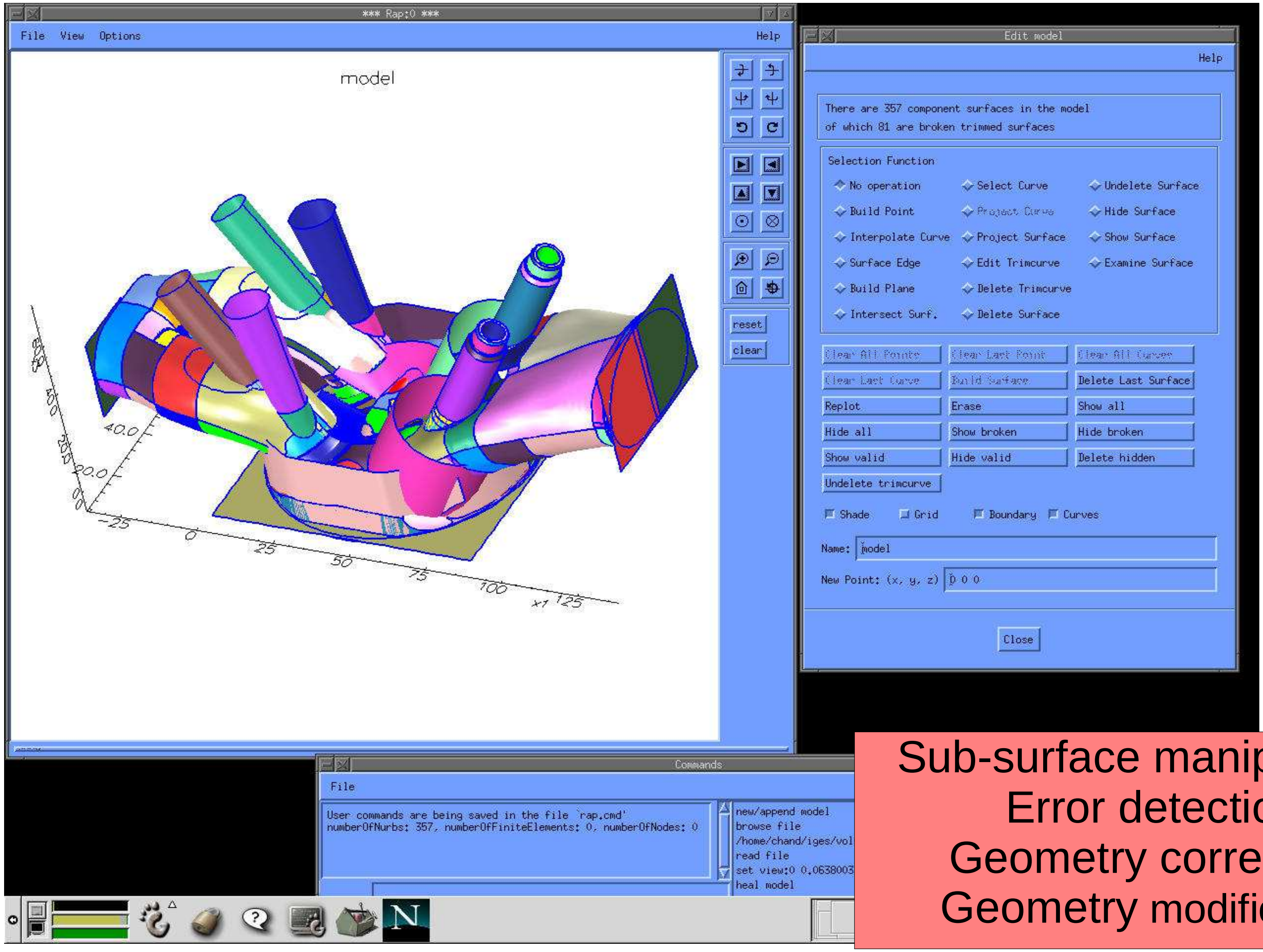








Rap: Composite Surface Editing



There are 357 component surfaces in the model
of which 81 are broken trimmed surfaces

Shade   Grid   Boundary   Curves

Name: model

New Point: (x, y, z) 0 0 0

```
User commands are being saved in the file `rap.cmd'
numberOfNurbs: 357, numberOfFiniteElements: 0, numberOfNodes: 0
```

```
new/append model
browse file
/home/chand/iges/vol
read file
set view:0 0.0638003
heal model
```

Sub-surface manipulation
Error detection
Geometry correction
Geometry modification

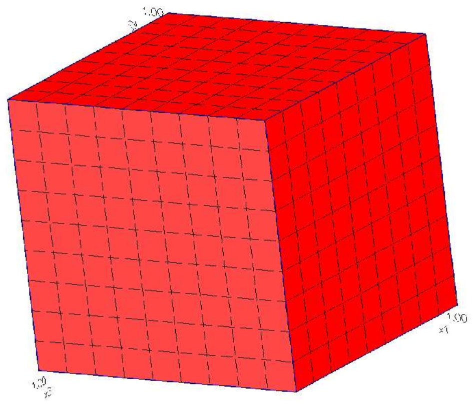
Using Mappings for geometry and mesh generation : specific examples

Example 0 : Overture/Rapsodi fundamentals
- library initialization
- creating a simple Mapping
- basic plotting

Example 1 : An interactive IGES reader and data query tool
- creating or reading a CompositeSurface
- use of map, inverseMap and project
- introduction to IntersectionMapping

Example 2 : A 2D unstructured mesh generator
- an application that uses Rapsodi within its own infrastructure
- grids as evaluations of Mappings
- UnstructuredMapping

Example 0: Build and plot a Mapping

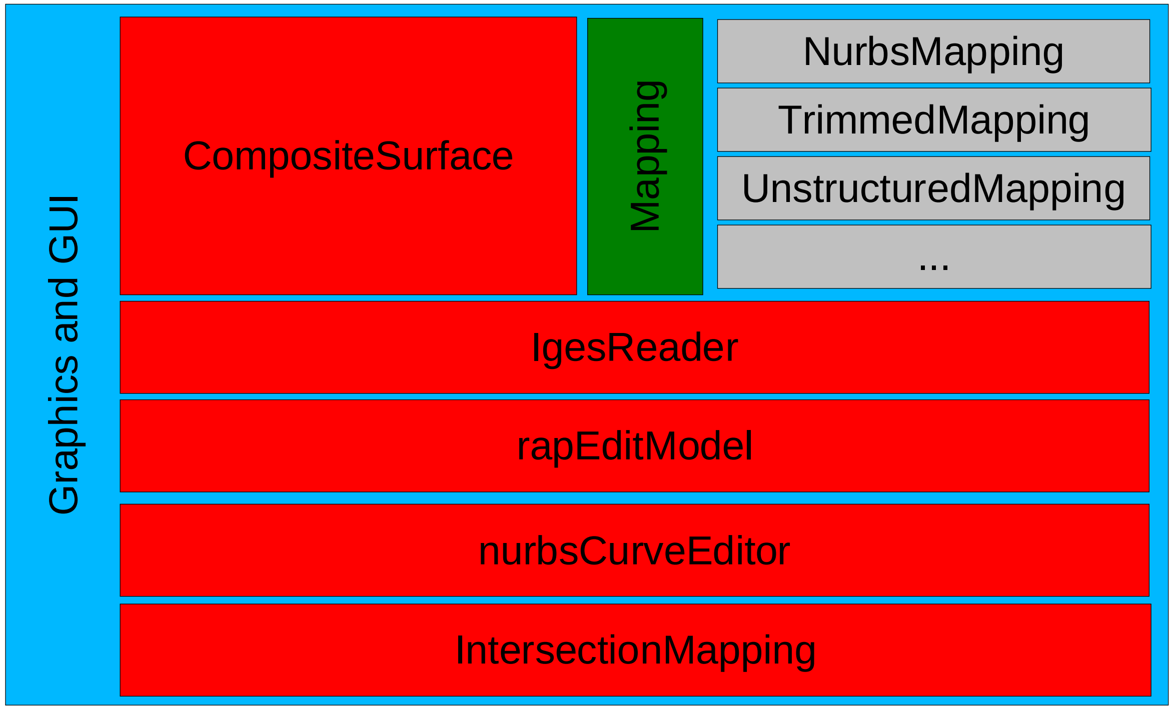


```
#include "GenericGraphicsInterface.h"
#include "PlotIt.h"
#include "BoxMapping.h"

int main(int argc, char *argv[])
{
   // initialize the Overture library
   Overture::start(argc,argv);

   // ask the library for a graphics interface
   GenericGraphicsInterface &gi =
                *Overture::getGraphicsInterface()
   GraphicsParameters gp;

   BoxMapping box;
   PlotIt::plot(gi,box,gp);

   // let the library clean up after itself
   Overture::finish();

   return 0;
}
```

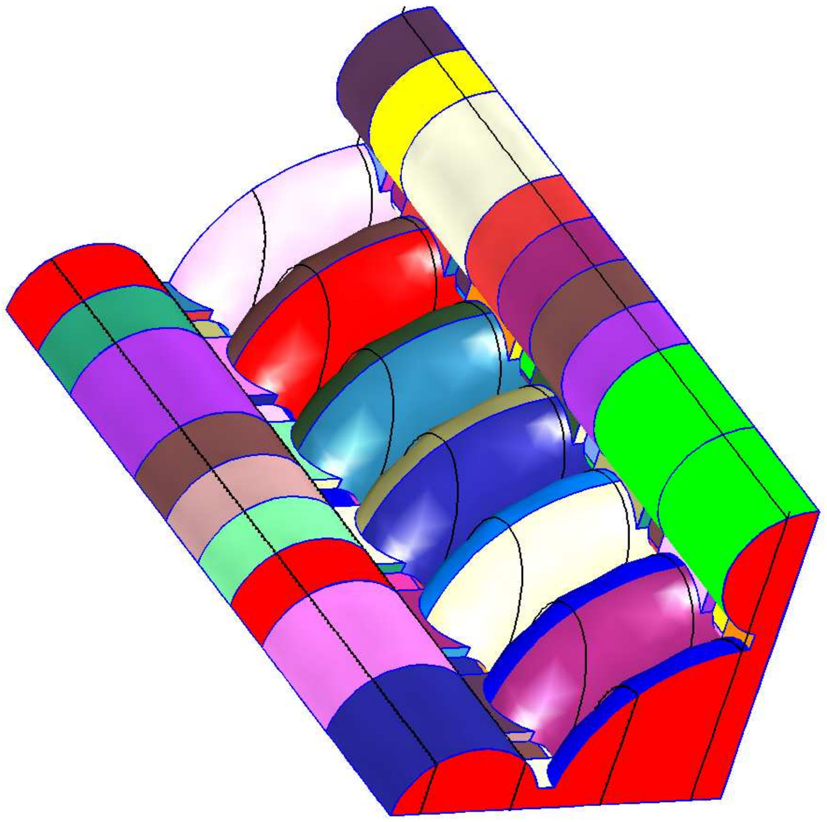
Example 1: IGES reader and query tool

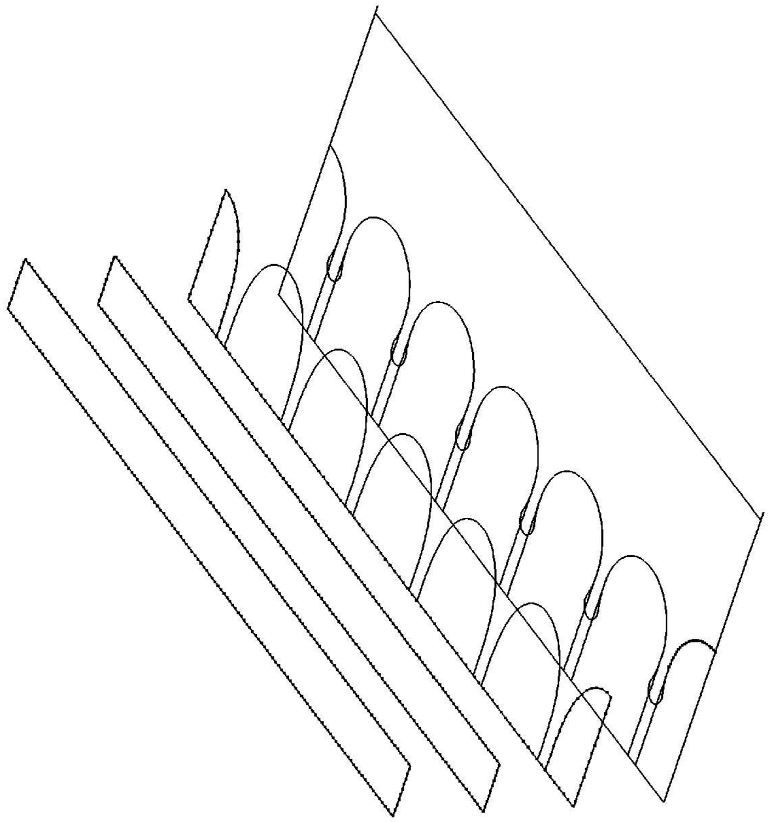
Read, view and edit IGES files
Query points on the model
Intersect the model with arbitrary planes
 -->edit and output the resulting curves

1200 lines of code, mostly gui



Example 1: Slice curve from an IGES model





Example 1: Code examples

Interactively read and plot an IGES file:

```
GenericGraphicsInterface & gi =
                Overture::getGraphicsInterface();
MappingInformation mapInfo;
CompositeSurface model;

if ( rapNewModel(gi, mapInfo, model) )
   PlotIt::plot(gi,model);
```

-- Or, without the GUI and graphics:

```
MappingsFromCAD cadReader;
IgesReader *igesReader=NULL;
int nNurbs, nFE, nNodes, status;

cadReader.fileContents("file.igs",igesReader, nNurbs,
                        nFE, nNodes,status);

model = cadReader.readSomeNurbs(mapInfo,
                  igesReader, 0, nNurbs, nNurbs, status);
```

Example 1: Code examples

Projecting points onto a CompositeSurface (or Mapping):

```
CompositeSurface model; // get model from somewhere
RealArray xToProject;
// fill in vertices to project...

MappingProjectionParameters mp;
model.project(xToProject,mp);
```

Intersect 2 Mappings using an IntersectionMapping

```
Mapping &map1 = someMapping; // surface in R3
Mapping &map2 = anotherMapping; // surface in R3
IntersectionMapping intersection;

intersection.intersect(someMapping, anotherMapping);

intersection.map(...);
intersection.project(...);
```

curve intersection is similar

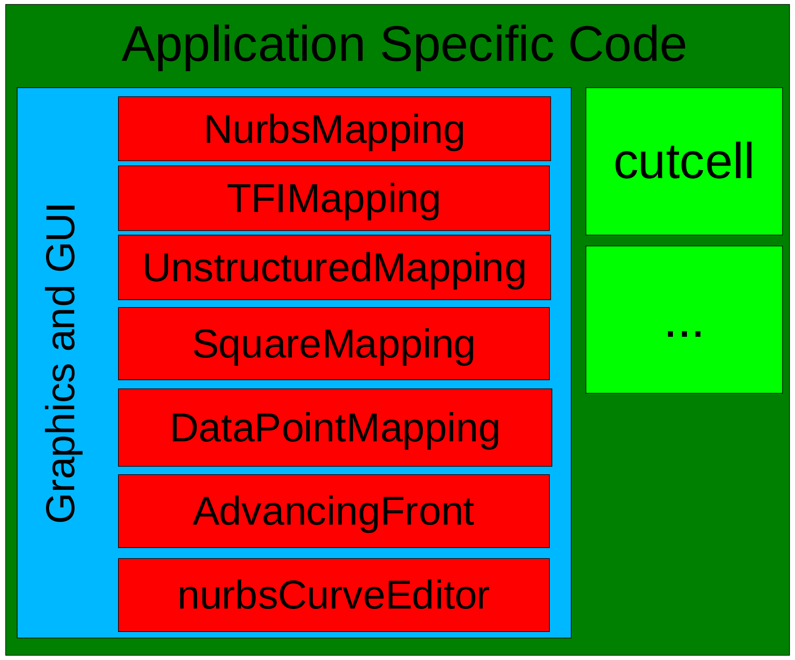
Example 2: Simple 2D mesh generator

Unstructured, multi-region 2D meshes
Uses three tools for mesh generation

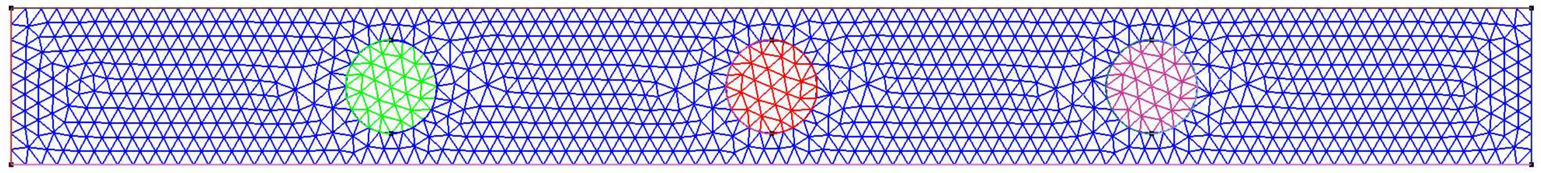
1. TFIMapping
2. AdvancingFront
3. User created cutcell + AdvancingFront

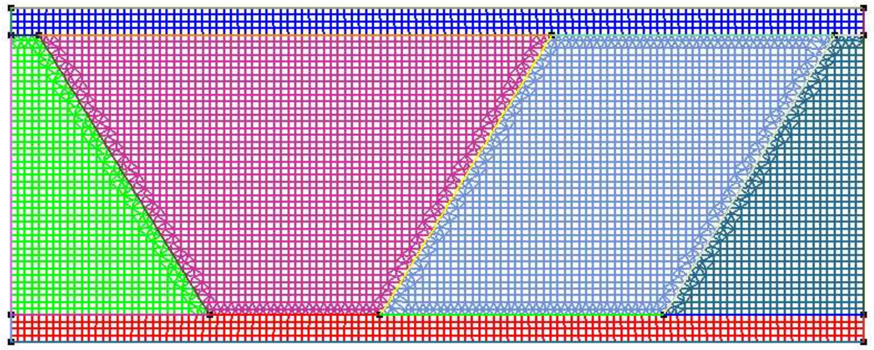
4048 lines of code:

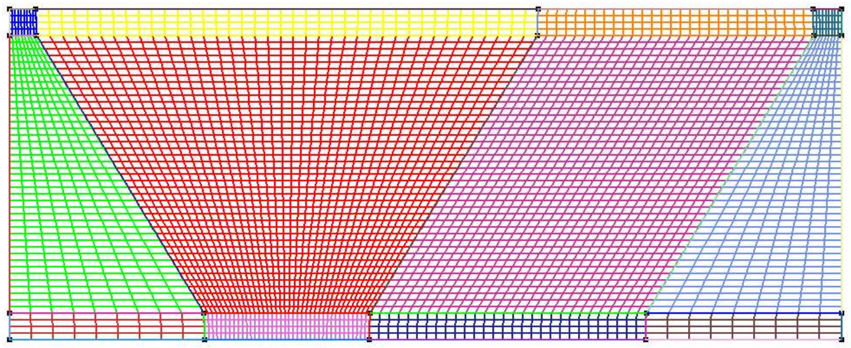
- 1388 for interactive gui/graphics/error handling (Overture GUI)
- 1812 for the "application" (uses Overture Mappings)
- 848 for the cutcell algorithm (no Overture code)

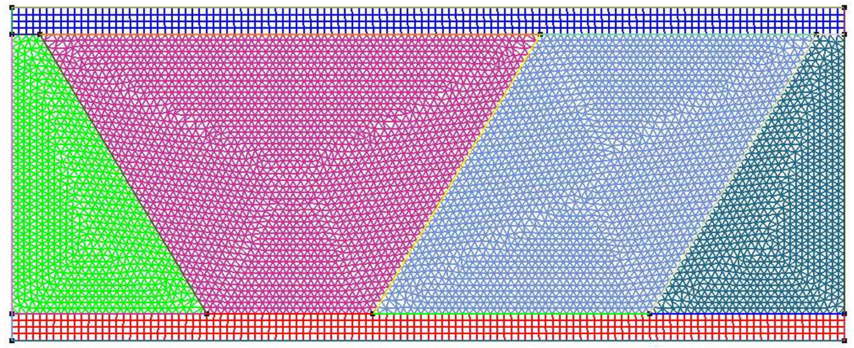


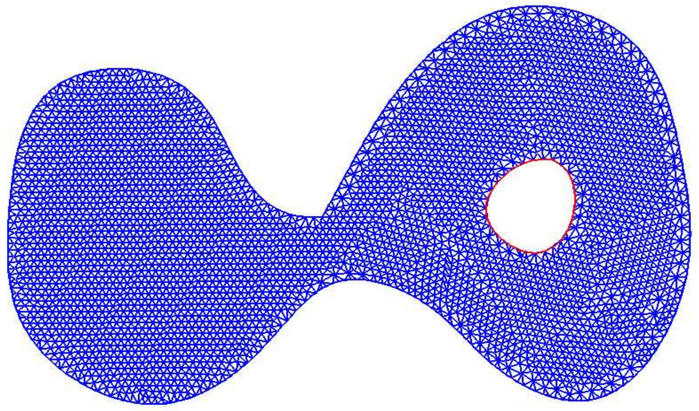
Example 2: Sample meshes



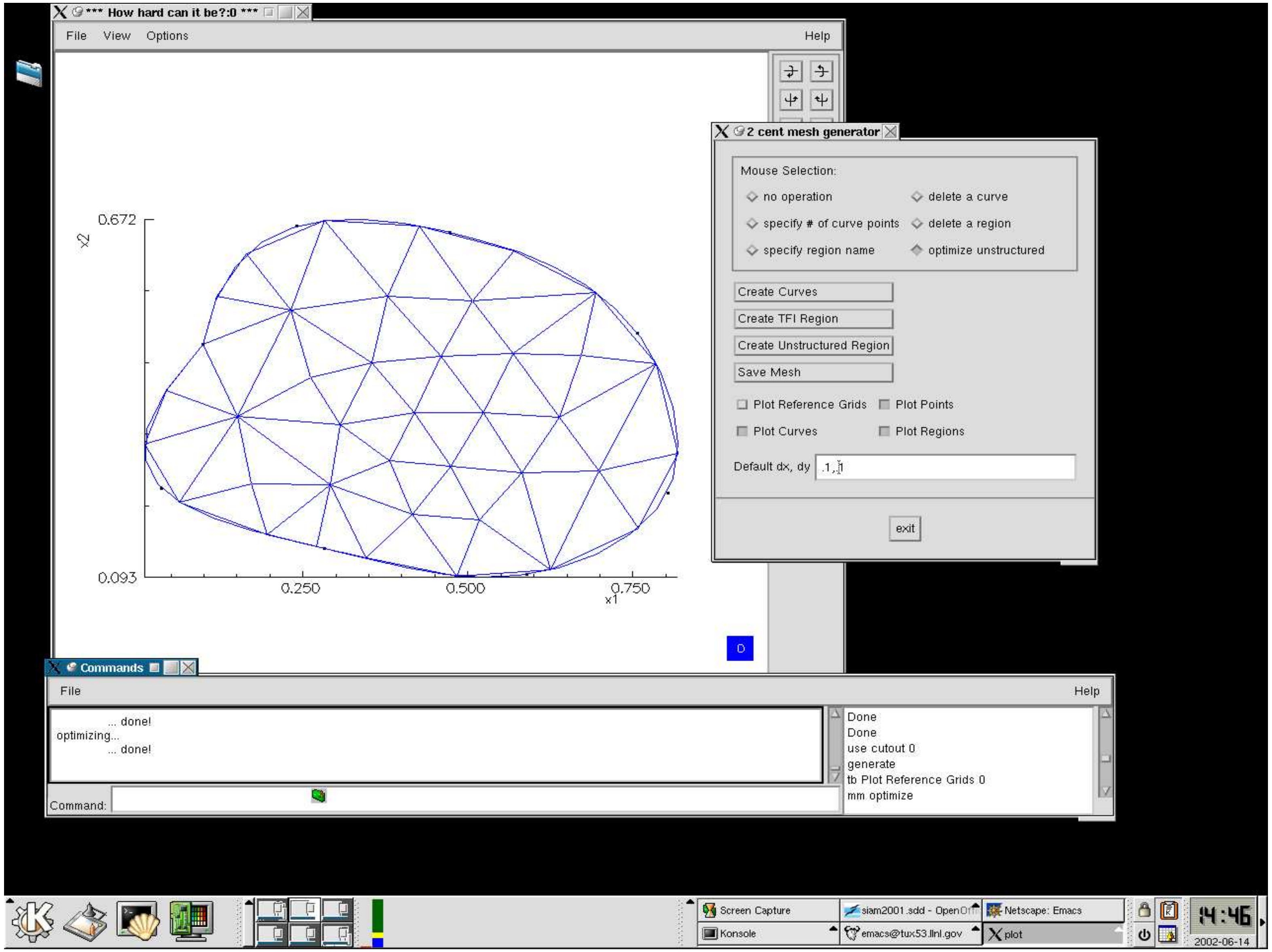








Example 2: Interactive interface



Default dx, dy .1,.1

```
            ... done!
optimizing...
            ... done!
```

Command:

```
Done
Done
use cutout 0
generate
tb Plot Reference Grids 0
mm optimize
```

Example 2: Code examples

Mappings can be evaluated to generate a grid

```
Mapping &map = someSpecificMapping;
RealArray xGrid;

// either use Mapping::getGrid
map.setGridDimension(axis, nX1);
xGrid = map.getGrid();

// or provide a grid in the domain to a specialized map
RealArray rGrid = ...;
map.mapGrid(rGrid,xGrid); // or map a list of vertices
```

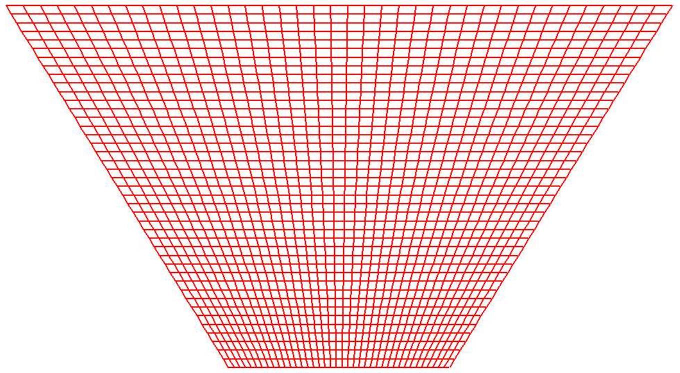
A grid of vertices in the range can also be inverted:

(useful for projecting patches onto surfaces)

```
Mapping &map = someSpecificMapping;
RealArray rGrid,xGrid;

map.inverseMapGrid(xGrid,rGrid); // or inverseMap
```

Example 2: Grid generation with Mappings

```
Mapping *left, *right, *bottom, *top;
//...
TFIMapping tfi;
int nX1, nX2;

tfi.setDomainDimension(2);
tfi.setRangeDimension(2);  // = 3 for a surface

// set the sides, note these are referenced!
// could add front/back too...
tfi.setSides(left,right,bottom,top);

tfi.setGridDimensions(axis1, nX1);
tfi.setGridDimensions(axis2, nX2);
RealArray &grid = tfi.getGrid();
```



Useful Mappings for grid generation:

BoxMapping	StretchMapping
HyperbolicMapping	ComposeMapping
CylinderMapping	RevolutionMapping
SweepMapping	DepthMapping

UnstructuredMapping

UnstructuredMapping is special:

- map/inverseMap not available
- grid size is immutable
- project is available
- supports 2 and 3D hybrid meshes
- can be read from ply, ingrid-style and IGES files
- can be written to Overture database and ingrid-style files
- currently has implicit connectivity based on arrays

```
class UnstructuredMapping : public Mapping
{
public:
   virtual int  project(...);

   int setNodesAndConnectivity(...); // + optimized versions
   int buildFromAMapping(...); // + optimized versions

   int findBoundaryCurves(...); // usefull for surfaces

   const RealArray & getNodes() const;
   const intArray & getElements() const;
   // ... + other connectivity and Mapping methods
};
```

Some useful utilities

Fast geometric search tree

Interfaces to Jonathan Shewchuk's robust predicates
    and delaunay triangulator

2 and 3D adaptive precision intersection functions
    line-line (2D) and line-triangle (3D)
    implementation uses Shewchuk's predicates

Interactive NURBS curve editor function (uses GUI)

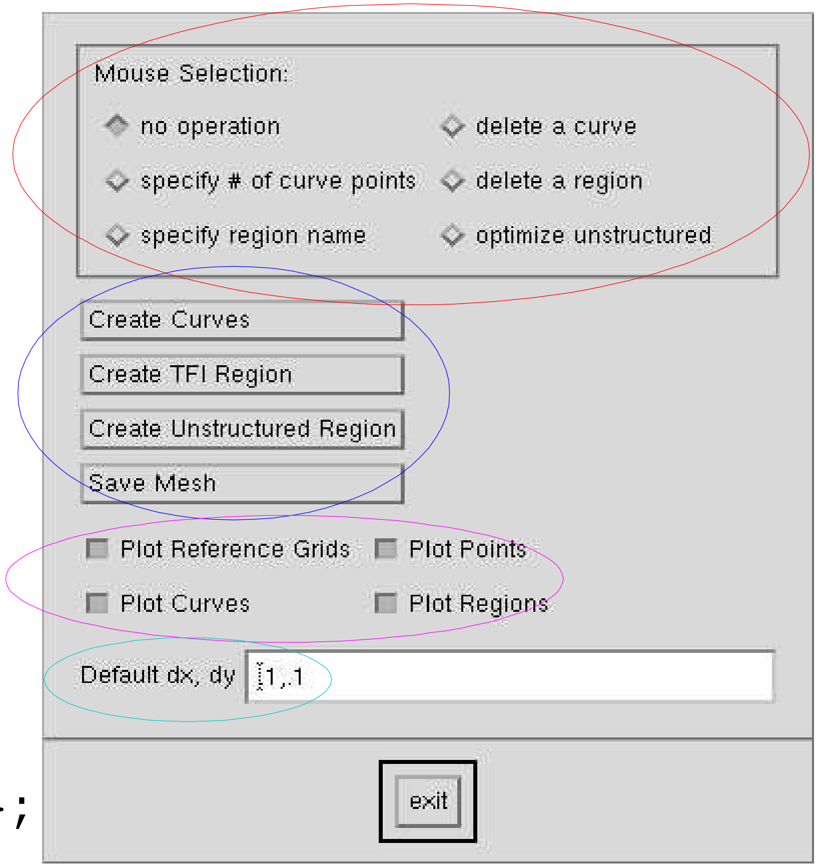
Interactive CompositeSurface editor function (uses GUI)

HDF4 based database interface

2 and 3D AdvancingFront mesh generator

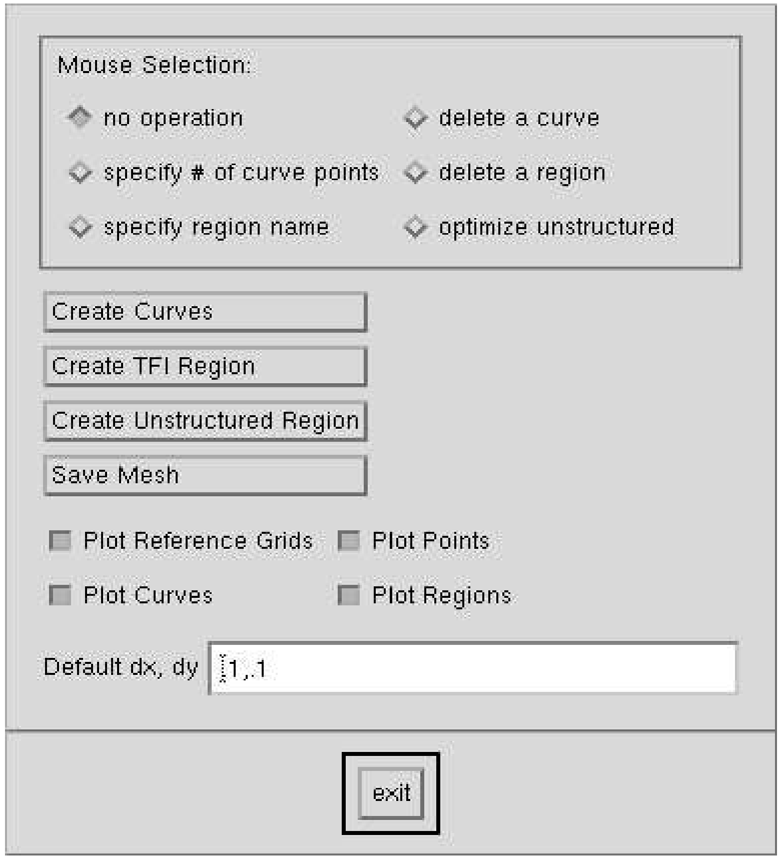
Graphics Interface : Setting up a GUI

```
GUIState gui;
gui.setWindowTitle("2 cent mesh generator");

aString pickCmd[] = {"mm noOp",...., ""};
aString pickLbl[] = {"no operation",
                     ...,""};
int defPick = 0;
gui.addRadioBox("Mouse Selection",
                pickCmd,pickLbl,
                defPick,nRows);

aString pbCmd[] = {...};
aStirng pbLbl[] = {...};
gui.setPushButtons(pbCmd,pbLbl,nRowsPb);

aString tbCmd[]={...};
aString tbLbl[] = {...};
int tbState[] = {true,true,true,true};
gui.setToggleButtons(tbCmd,tbLbl,
                     tbState,nRows);

aString txtLbl[] = {"Default dx, dy",""};
aString txtCmd[] = {"dxdy", ""};
aString txtInit[]= {".1",".1",""};
gui.setTextBoxes(txtCmd,txtLbl,txtInit);

gui.setExitCommand("exit","exit");
```


Mouse Selection:
◆ no operation ◇ delete a curve
◇ specify # of curve points ◇ delete a region
◇ specify region name ◇ optimize unstructured

Graphics Interface : Using a GUI

```
GenericGraphicsInterface &gi = ...;

SelectionInfo select;
aString answer;

gi.pushGUI(gui);
while(1)
{
   // blocks until something happens
   gi.getAnswer(answer, "", select);

   if ( answer.matches("exit") )
      break;
   else if ( answer.matches(...) )
      //...
   else if ( select.nSelect )
      // ...
   else
      {
         aString msg =
            "unknown command : "+answer;
         gi.outputString(msg);
         gi.createMessageDialog(msg, errorDialog);
      }
}
gi.popGUI();
```



Graphics Interface : GUI comments

GUI if/elseif blocks take up lines of code, but are generally
not too complicated.

GenericGraphicsInterface maintains a *stack* of GUIs. There is no
omnipotent outer event loop. getAnswer serves as the
local event loop.

The user never sees the underlying GUI implementation; currently
the underlying code uses MOTIF, but is limited to one file...

GenericGraphicsInterface may not even use graphics! It can
operate in a purely text mode

GenericGraphicsInterface :
- also handles the reading/writting of command, log and hardcopy files
- getAnswer intercepts certain inputs such as
  - rotation/translation/zoom, clipping, view parameters...
- non-blocking getAnswer enables the interruption of long
  - computations

Caveat Emptor

There are some bugs and idiosyncrasies

Development model is informal:
    Reasonable but not Rigorous

Development is moving to Linux:
    open issues include remote OpenGL/X11 performance

While developed for our research applications, the library is also useful for rapid prototyping

Obtaining Overture

Overture home page:
www.llnl.gov/CASC/Overture